Capstone Project - 2

Bike Sharing Demand Prediction

By- Raushan Kumar

Index

Problem Statement

- Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. It is important to make the rental bike available and accessible to the public at the right time as it lessens the waiting time. Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.

Goal

- The goal of our project was to predict the bike count required at each hour for the stable supply of rental bikes.
- As It is important to make the rental bike available and accessible to the public at the right time as it lessens the waiting time

Data Description

The dataset contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour and date information.

Attribute Information:

- Date : year-month-day
- Rented Bike count - Count of bikes rented at each hour
- Hour - Hour of he day
- Temperature-Temperature in Celsius
- Humidity - %
- Windspeed - m/s
- Visibility - 10m
- Dew point temperature - Celsius
- Solar radiation - MJ/m2
- Rainfall - mm
- Snowfall - cm
- Seasons - Winter, Spring, Summer, Autumn
- Holiday - Holiday/No holiday
- Functional Day - NoFunc(Non Functional Hours), Fun(Functional hours)

Data Description

	Rented_Bike_Count	Hour	Temperature	Humidity	Wind_speed	Visibility	Dew_point_temperature	Solar_Radiation	Rainfall	Snowfall
count	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000
mean	704.602055	11.500000	12.882922	58.226256	1.724909	1436.825799	4.073813	0.569111	0.148687	0.075068
std	644.997468	6.922582	11.944825	20.362413	1.036300	608.298712	13.060369	0.868746	1.128193	0.436746
min	0.000000	0.000000	-17.800000	0.000000	0.000000	27.000000	-30.600000	0.000000	0.000000	0.000000
25%	191.000000	5.750000	3.500000	42.000000	0.900000	940.000000	-4.700000	0.000000	0.000000	0.000000
50%	504.500000	11.500000	13.700000	57.000000	1.500000	1698.000000	5.100000	0.010000	0.000000	0.000000
75%	1065.250000	17.250000	22.500000	74.000000	2.300000	2000.000000	14.800000	0.930000	0.000000	0.000000
max	3556.000000	23.000000	39.400000	98.000000	7.400000	2000.000000	27.200000	3.520000	35.000000	8.800000

```
# Quick info of our given dataset
df_seoul.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Date                   8760 non-null   object
 1   Rented_Bike_Count      8760 non-null   int64
 2   Hour                   8760 non-null   int64
 3   Temperature            8760 non-null   float64
 4   Humidity               8760 non-null   int64
 5   Wind_speed             8760 non-null   float64
 6   Visibility             8760 non-null   int64
 7   Dew_point_temperature  8760 non-null   float64
 8   Solar_Radiation        8760 non-null   float64
 9   Rainfall               8760 non-null   float64
 10  Snowfall               8760 non-null   float64
 11  Seasons                8760 non-null   object
 12  Holiday                8760 non-null   object
 13  Functioning_Day        8760 non-null   object
dtypes: float64(6), int64(4), object(4)
memory usage: 958.2+ KB
```

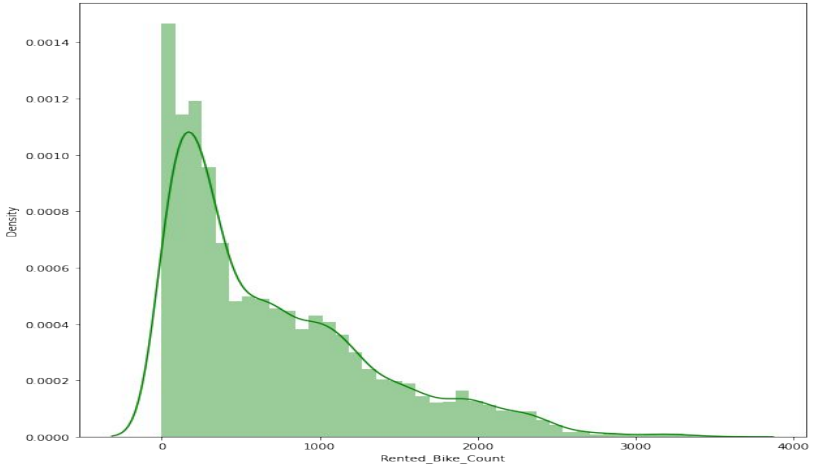
1. Their total 8760 rows and 14 columns.
2. Object data types columns - 4
3. Integer data types columns - 4
4. Float data types columns - 6
5. Unique count for Seasons, Holidays, Functioning Day is very very low like 4, 2 and 2 respectively.

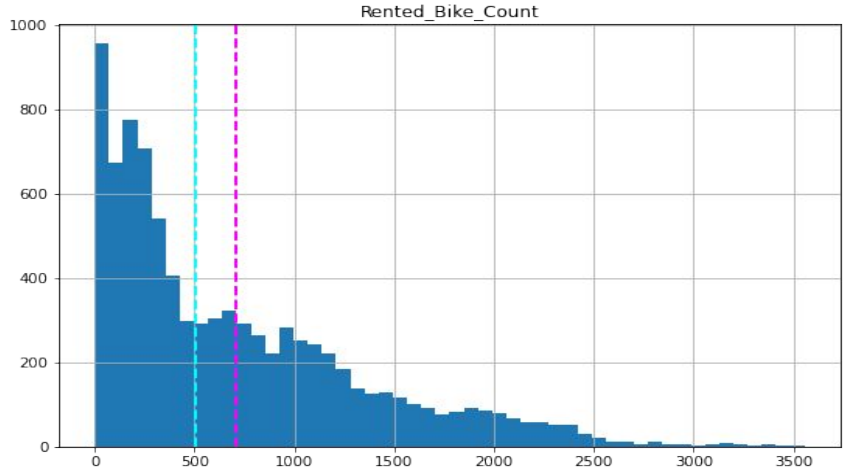
Data Wrangling

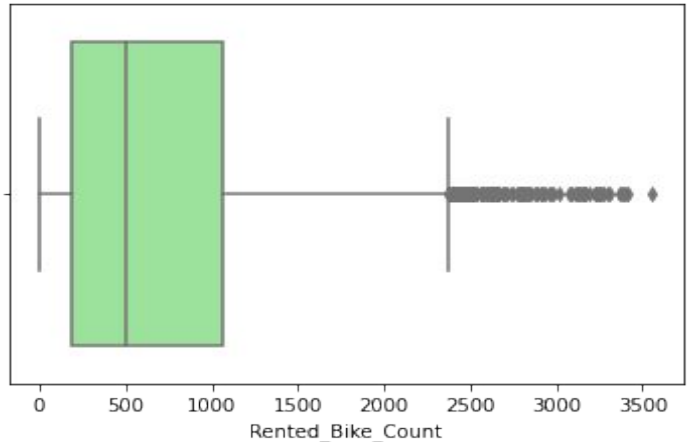
- Checking for duplicates : **None Found**
- Checking for NaN Values : **None Found**
- Our data is very clean.Their no any null values.
- No need of data cleaning step.

	Datatypes	Count of non-null values	NaN values	% NaN Values	Unique_count
Date	object	8760	0	0.0	365
Rented_Bike_Count	int64	8760	0	0.0	2166
Hour	int64	8760	0	0.0	24
Temperature	float64	8760	0	0.0	546
Humidity	int64	8760	0	0.0	90
Wind_speed	float64	8760	0	0.0	65
Visibility	int64	8760	0	0.0	1789
Dew_point_temperature	float64	8760	0	0.0	556
Solar_Radiation	float64	8760	0	0.0	345
Rainfall	float64	8760	0	0.0	61
Snowfall	float64	8760	0	0.0	51
Seasons	object	8760	0	0.0	4
Holiday	object	8760	0	0.0	2
Functioning_Day	object	8760	0	0.0	2

Exploratory View – Dependent Variable

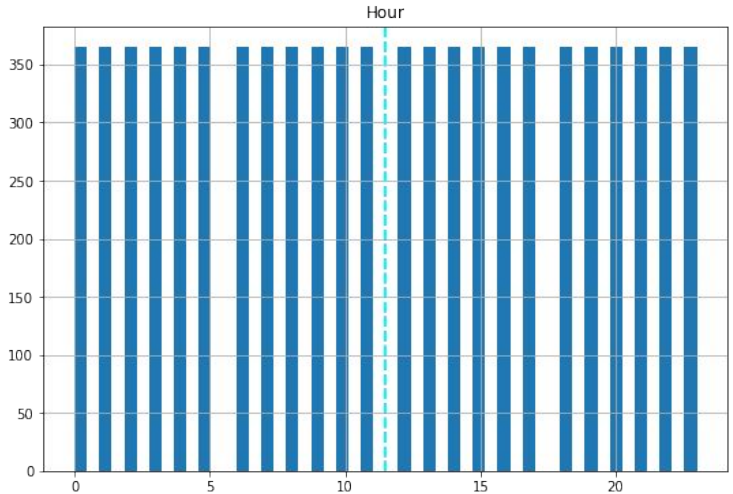


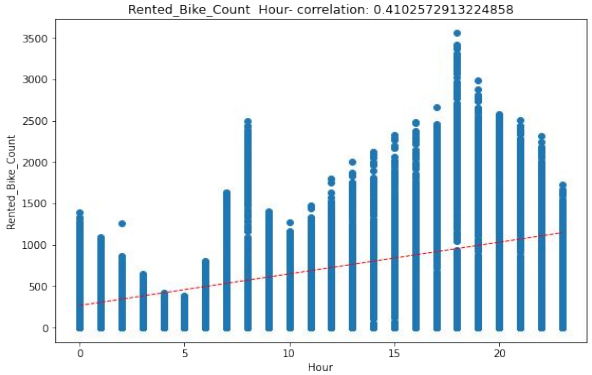




1. Most of the Rented bike count lies between 0 and 2300.
2. Distribution is positively skewed, not normal
3. And most count is 500.

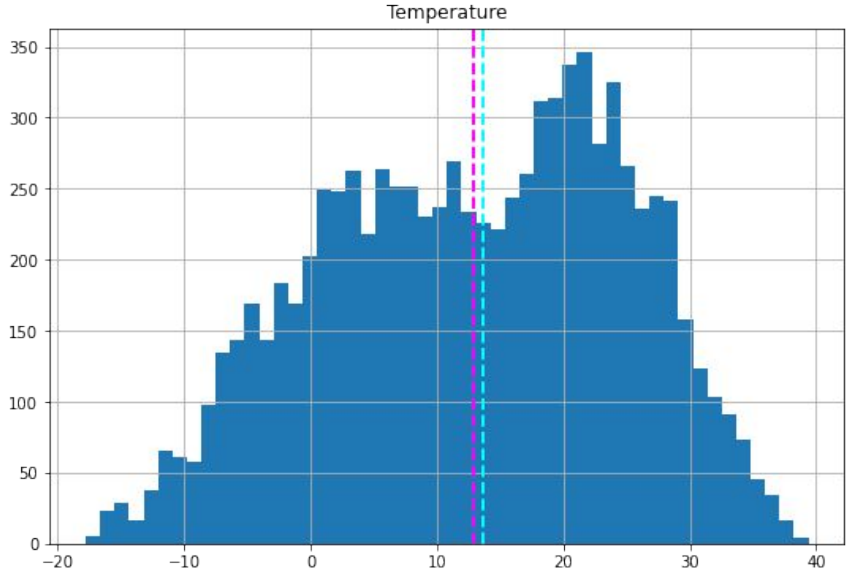
Feature Variable – Hour Analysis

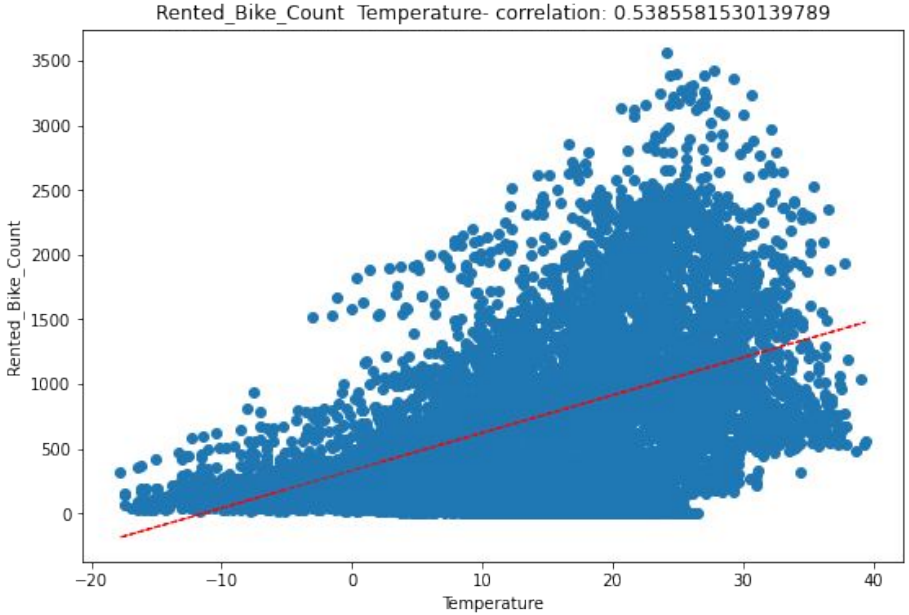




- Scatter plot showing most of the high demand during office timings around 8 A.M. and 8 P.M.
- And for 4 A.M. and 5 A.M. demand is very low.
- From all we can say timing is a important feature for predicting demand.

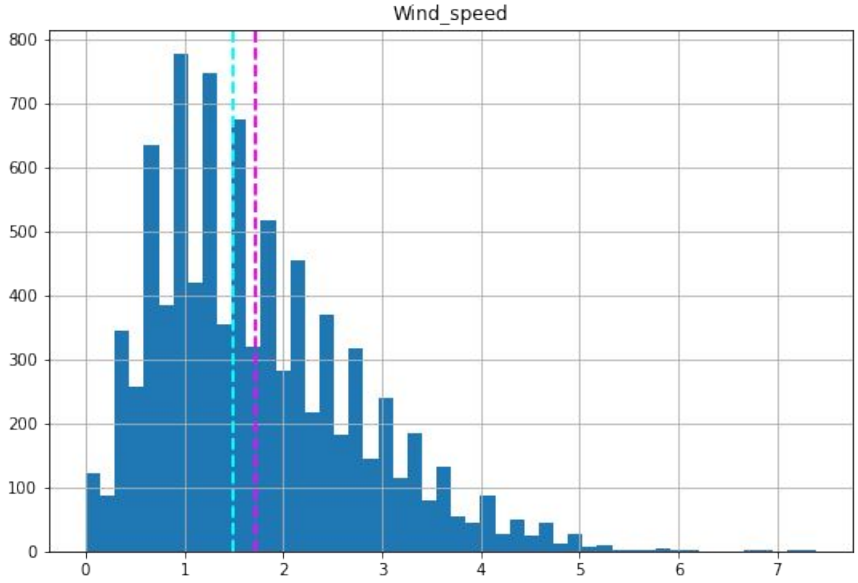
Feature Variable – Temperature Analysis

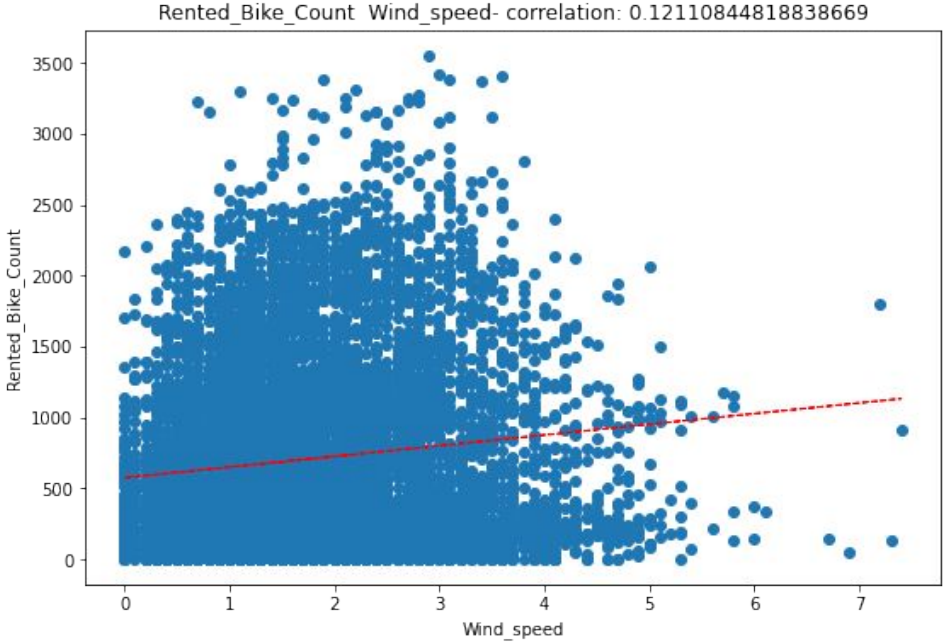




- This Plot shows demand is increasing with increasing temperature.

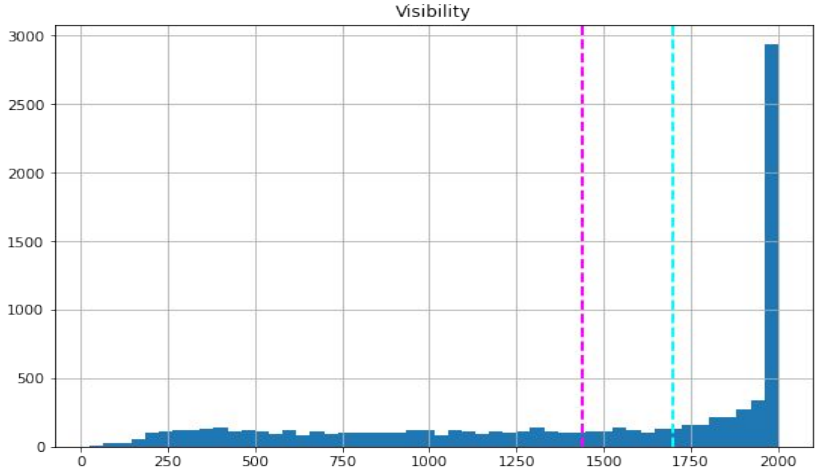
Feature Variable – Wind Speed Analysis

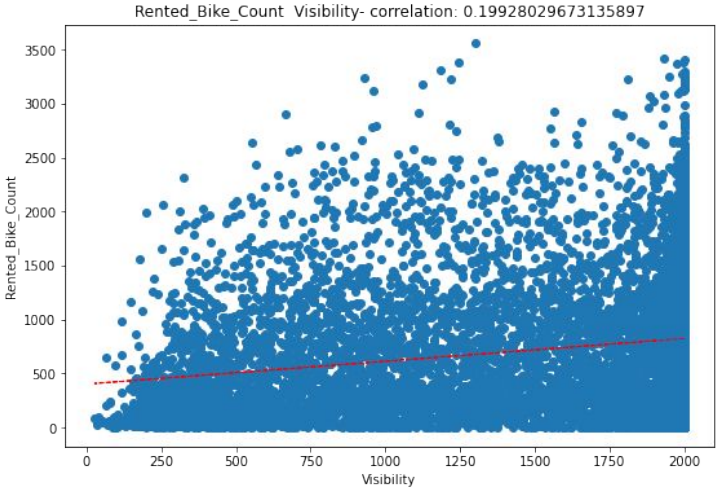




- From here we can infer that the Wind scatter plot shows an increase in wind speed and slight increase in demand.

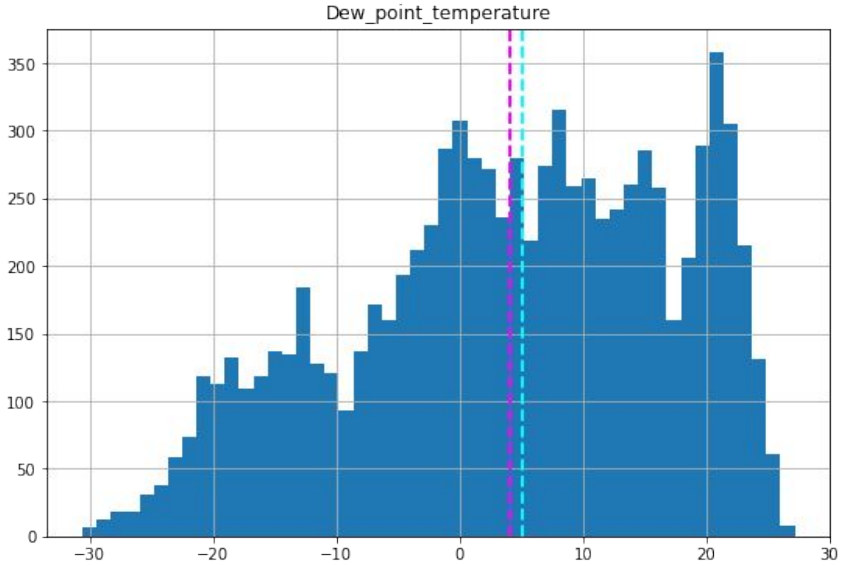
Feature Variable – Visibility Analysis

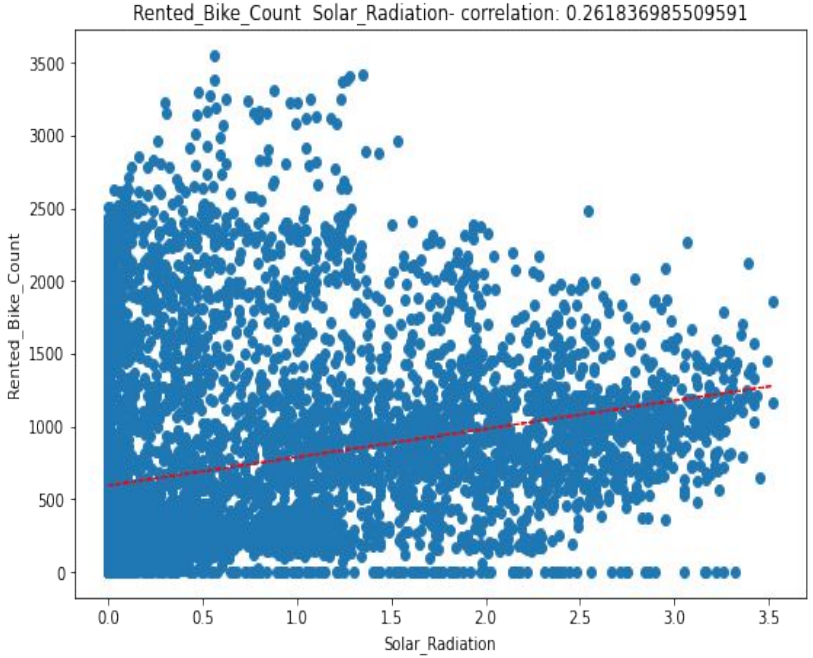




- We can infer that Visibility have approx 20% correlation with rented bike demand, so as visibility increases demand also increases.

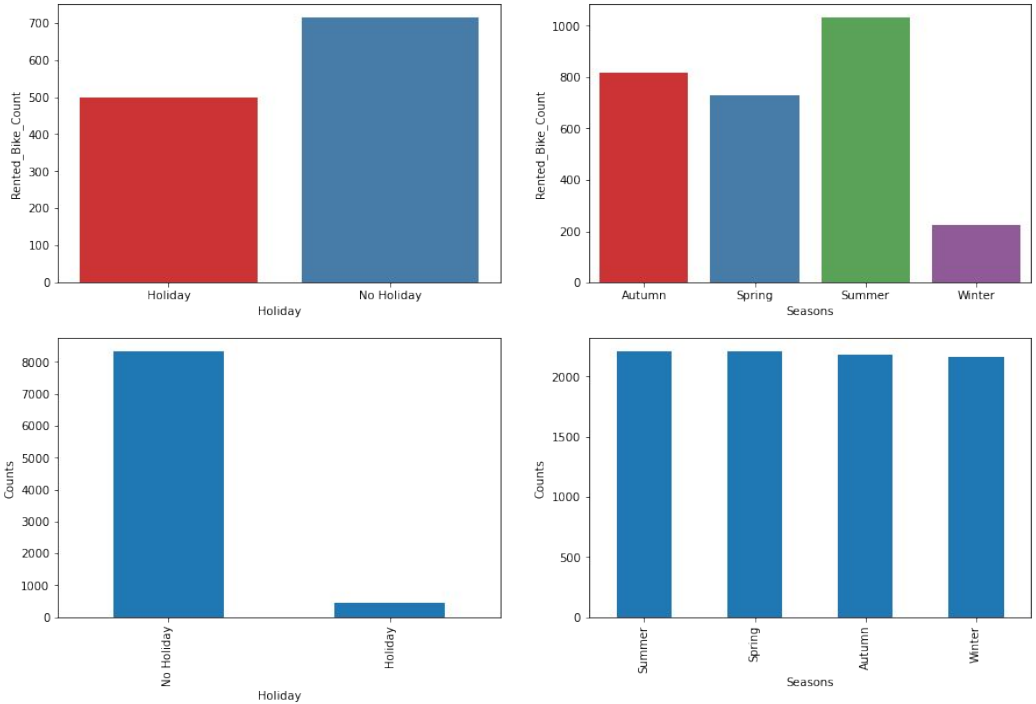
Feature Variable – Solar radiation Analysis





- From here we can see With increasing in solar radiation bike demand increases but slightly.

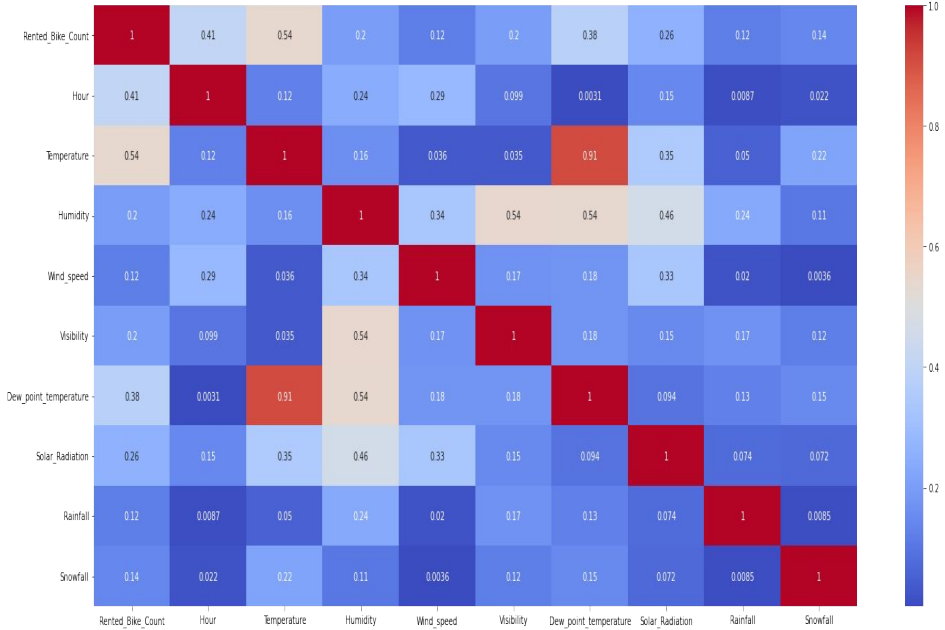
Feature Variable – Categorical Variable Analysis



From categorical features we can infer that **Most of the bikes are rented during working days or on no holiday**

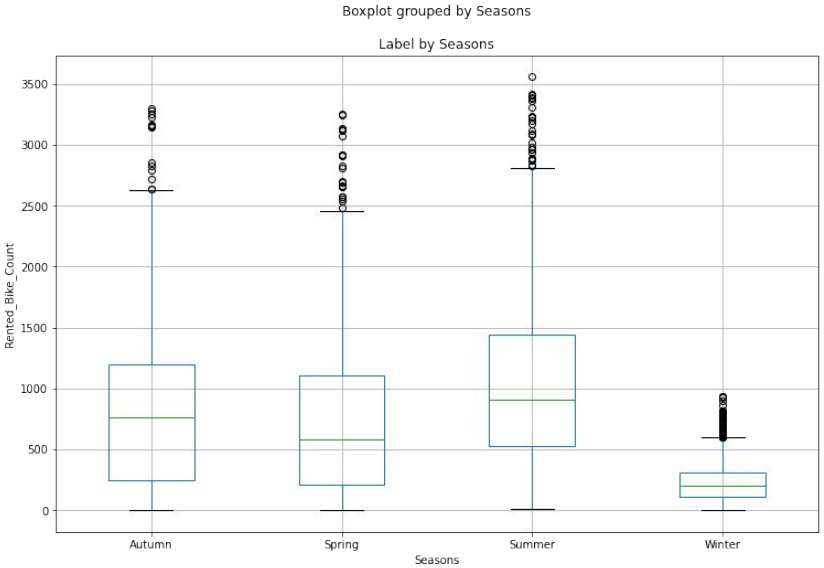
- we also observe that people prefer renting bikes in the summer season followed by autumn season.

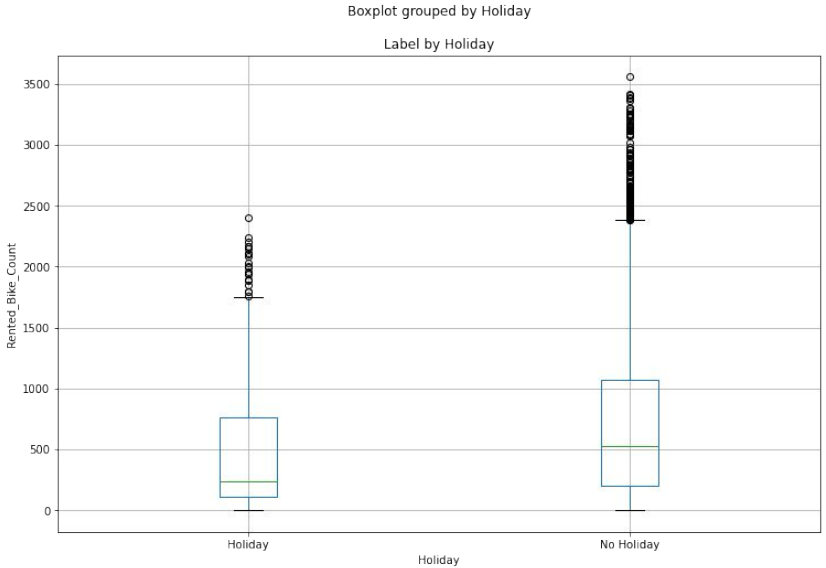
Correlation Matrix



1. From the correlation graph we clearly observe that the features related to temperature and features related to dew point temperature have positive correlation within themselves i.e. 91%.
2. Rainfall and snowfall have very low relations.
3. Temperature and rented bike count have very good correlation between them.

Outlier





- Most of the bike rented during working days or on no holiday.
- We also observe that people prefer more renting bikes in summer season followed by autumn season.

Data Splitting & Models Used

```
# Data for all the independent variables
independent_variables = df_seoul.drop(labels='Rented_Bike_Count',axis=1)

# Data for the dependent variable
dependent_variable = df_seoul['Rented_Bike_Count']
```

```
# Create the data of independent variables
X = independent_variables.copy()

# Create the dependent variable data
y = dependent_variable.copy()
```

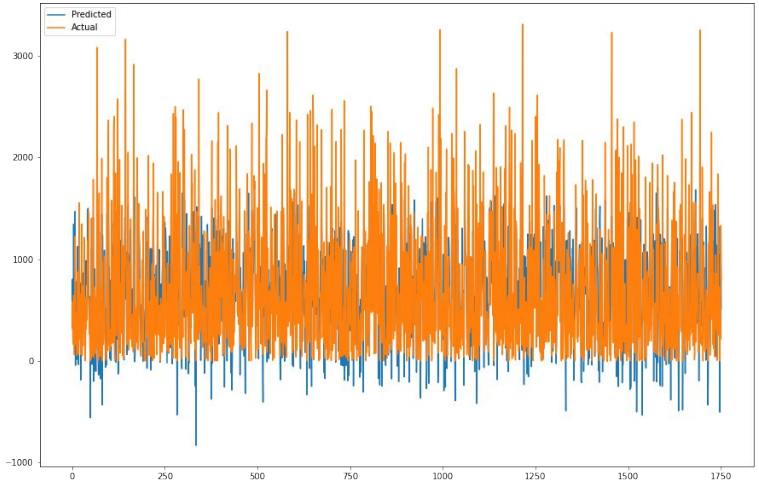
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2, random_state= 0)
print(X_train.shape)
print(X_test.shape)
```

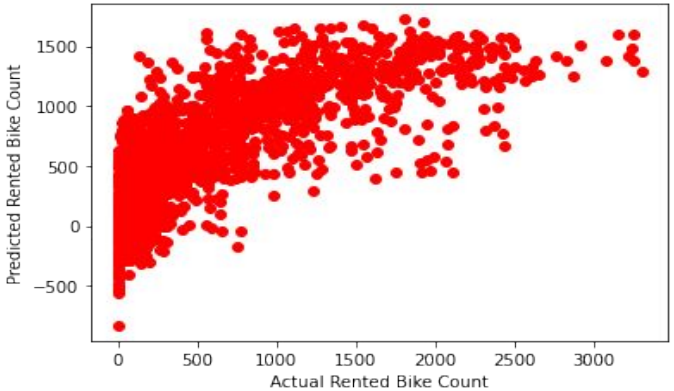
- Linear Regression
- Lasso Regression
- Ridge Regression
- Random Forest Regression
- XGBoost Regression
- Cross Validation on XGBoost
- Hyperparameter tuning

Performance of Model- Linear Regression

```
#Evaluation metrices
print("Adjusted R2 : ",1-(1-r2_score((y_test)
MSE  = mean_squared_error(y_test, y_pred)
print("MSE :" , MSE)
RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
print("MAE :" , MAE)
```

```
Adjusted R2 :  0.5403021020933562
MSE : 191075.43475018447
RMSE : 437.1217619270224
MAE : 325.8907133235788
```






```
r2_score(y_test, y_pred)
```

```
0.5434525160139043
```

- In linear regression we obtained the r2 score 0.54.

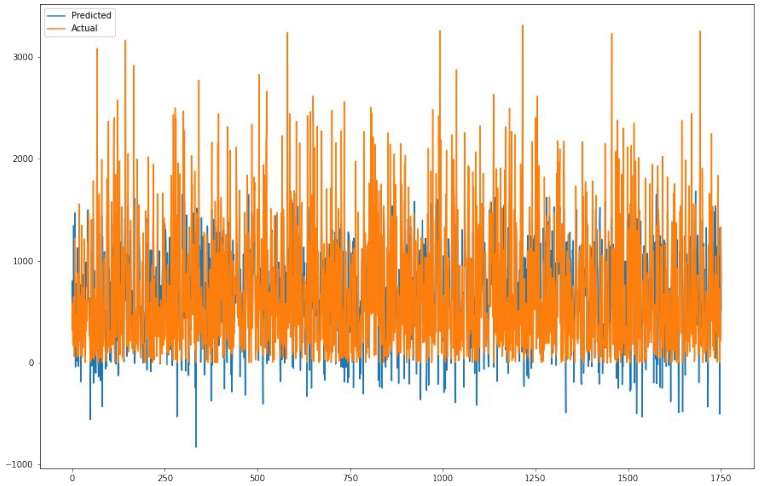
Performance of Model- Lasso Regression

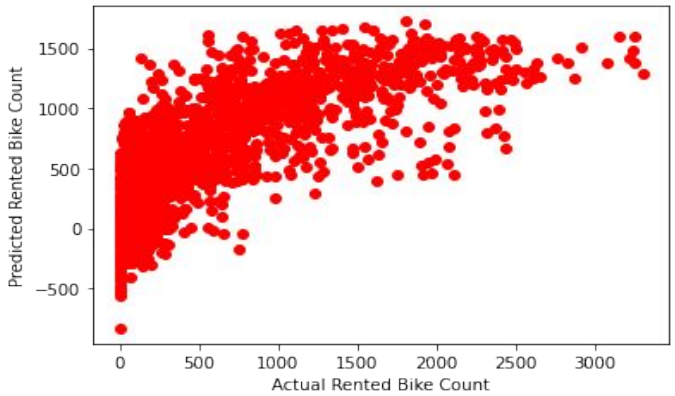
```
r2_score(y_test, y_pred_lasso)
```

```
0.5434403690036465
```

```
#Evaluation metrices
print("Adjusted R2 : ",1-(1-r2_score((y_test
MSE  = mean_squared_error(y_test, y_pred_las
print("MSE :" , MSE)
RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred_la
print("MAE :" , MAE)
```

```
Adjusted R2 :  0.540289871262441
MSE : 191080.51854835954
RMSE : 437.12757697079644
MAE : 325.88821142615546
```



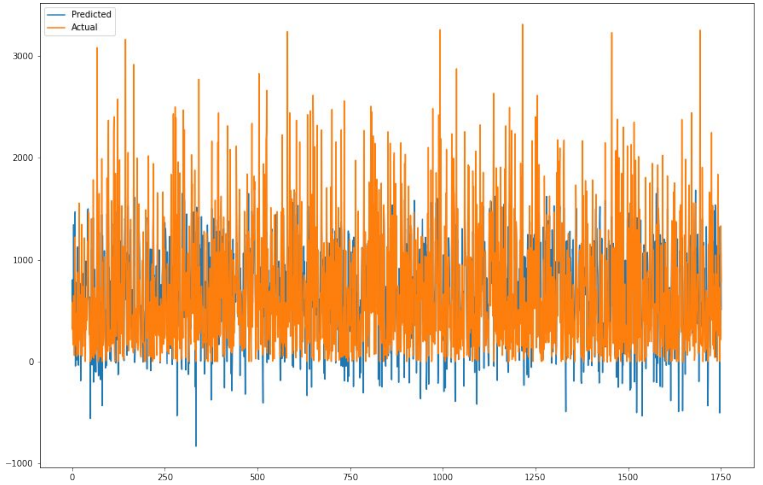


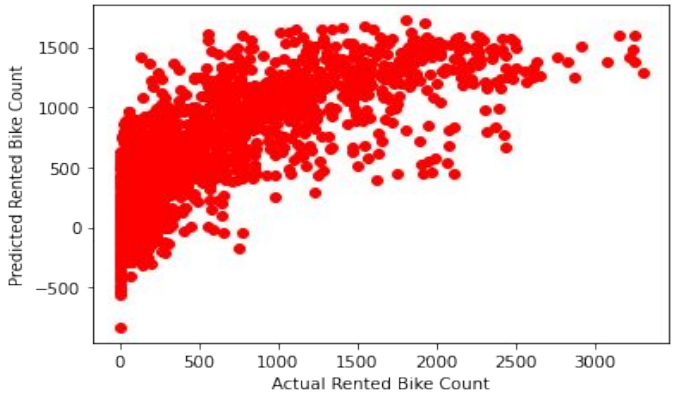
Performance of Model- Ridge Regression

```
[ ]  r2_score(y_test,y_predict_r)

     0.5434526142573535
```

```
#Evaluation metrices
print("Adjusted R2 : ",1-(1-r2_score((y_test)
MSE  = mean_squared_error(y_test, y_predict_r
print("MSE :" , MSE)
RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_predict_
print("MAE :" , MAE)
```

```
Adjusted R2 :  0.540302201014736
MSE : 191075.39363308184
RMSE : 437.1217148953845
MAE : 325.8893701288135
```





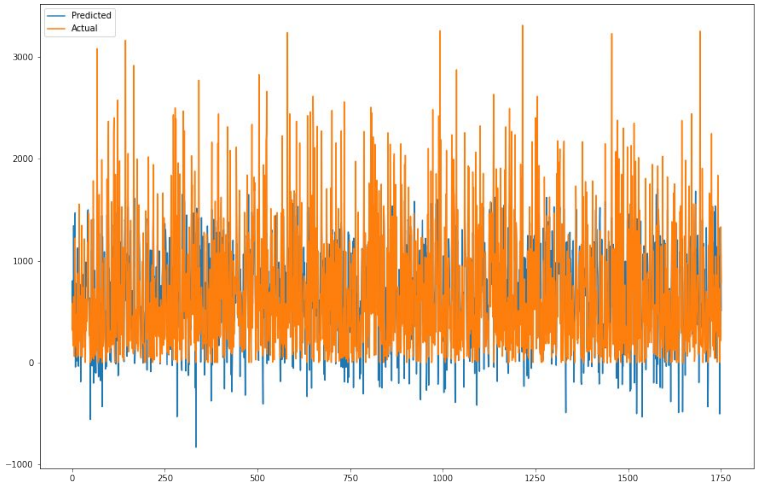
Performance of Model- Random Forest Regression

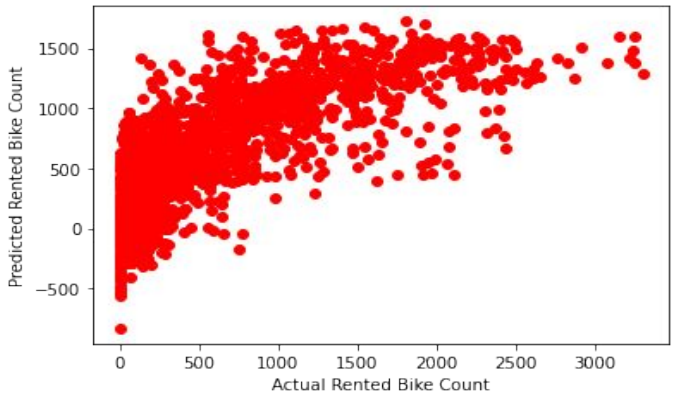
```
r2_score(y_test,y_predict_r)
```

```
0.5434526142573535
```

```
#Evaluation metrices
print("Adjusted R2 : ",1-(1-r2_score((y_test), (y
MSE  = mean_squared_error(y_test, y_predict_r)
print("MSE :" , MSE)
RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_predict_r)
print("MAE :" , MAE)
```

```
Adjusted R2 :  0.540302201014736
MSE : 191075.39363308184
RMSE : 437.1217148953845
MAE : 325.8893701288135
```





Performance of Model- XGBoost Regression

```
#Find R-squared value
r2 = r2_score(y_test, y_pred_xg)
r2
```

```
0.8786170470356018
```

```
# Find Adjusted R-squared value
print("Adjusted R2 : ",1-(1-r2_score((y_test), (y_pred_x
MSE  = mean_squared_error(y_test, y_pred_xg)
print("MSE :" , MSE)
RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred_xg)
print("MAE :" , MAE)
```

```
Adjusted R2 :  0.8777794418397578
MSE : 50801.507668893275
RMSE : 225.3918979663938
MAE : 140.27660047245897
```

From above we can infer that XGBoost have the highest r2 score 87.86% while other regression technique has only near to 54%. so we going to use this model for deployment.

```
train_score = dreg.score(X_train, y_train)
test_score = dreg.score(X_test,y_test)
print(f'Train score: {train_score}')
print(f'Test score: {test_score}')
```

```
Train score: 0.9667818524305548
Test score: 0.8786170470356018
```

XGBoost overfit our data on the training set as we can see from the above output.

Performance of Models – After Hyperparameter Tuning

For Test dataset:

```
#Find R-squared value
r2 = r2_score(y_test, test_preds)
# Find Adjusted R-squared value
adj_r2=1-(1-r2_score(y_test, test_preds))*((X_test.shape
```

```
r2
```

```
0.9520762741518607
```

```
adj_r2
```

```
0.9517455756411202
```

For Train dataset

```
#Find R-squared value
r2 = r2_score(y_train, train_preds)
# Find Adjusted R-squared value
adj_r2=1-(1-r2_score(y_train, train_preds))*(
```

```
r2
```

```
0.9514125490935283
```

```
adj_r2
```

```
0.9513291967831813
```

- The above output shows score on Test data and on both data test and training it performs very well.

- Test set R2 score is 0.95 improvements over 0.87 achieved using the untuned model.

Comparison of Performance of all the models

1. Best results over test set are given by XGBoost Regression with R2 score of 0.87.
2. Least RMSE score is also by Random Forest 0.58.
3. Lasso regularization over Linear regression was worst performing model.
4. As we can see XGBoost Regression it is showing that the R2 score for the train set is 96.67% but for the test it is 87.86% which shows that our model is overfitting on training data so to overcome this we did hyper parameter tuning for Extra Tree Regression.
5. And in last we successfully handle overfitting by optimizing parameters.

Conclusion

1. Most of the high demand **during office timings** around 8 A.M. and 8 P.M., and for **4 A.M. and 5 A.M. demand is very low**. from all we can say **timing is a important feature** for predicting demand.
2. We know that **demand is increasing with increasing temperature**.
3. And **demand is decreasing with increasing humidity**.
4. With **increase in wind speed their slightly increase in demand.**
5. Visibility have approx 20% correlation with rented bike demand, so as **visibility increases demand also increases.**
6. Dew point temperature has 38% correlation with demand, so as **DPT increases demand increases.**
7. With **increasing solar radiation bike demand increases but slightly**.
8. Generally **people don't prefer riding in rainfall and snowfall**, so they have very high negative correlation with bike demand.
9. Most of the **bike rented during working days or on no holiday**.
10. We also observe that **people prefer more renting bikes in summer season** followed by autumn season.

Scope of Improvement

Future work includes adding more predictive parameters and **Training other machine models such as ANN (Artificial Neural Network)** can further boost the predictive capacity; Seoul Bike Sharing is a improving domain and has a lot of reach in future. Also we can work on the day/week feature to explore more on the model performance, and try **various hyper-parameter tuning methods**.

Thanks